MongoDB com Java e Python

Fernando Anselmo

2020, MongoDB com Java e Python

Atualmente muito se tem comentado sobre bancos de dados não relacionais, também chamados de NoSQL. O conhecimento destes podem abrir várias portas e deve ser considerado um fator de extrema importância para garantir uma boa empregabilidade.É sempre importante estar atento a novas tecnologias e como elas resolvem problemas provenientes das limitações existentes no caso deste tipo de banco enormes quantidade de dados.

MongoDB com Java e Python

Fernando Anselmo
http://fernandoanselmo.orgfree.com/wordpress/

Versˆao 1.3 em 24 de outubro de 2021

Resumo

Atualmente muito se tem comentado sobre bancos de dados não relacionais, também chamados de NoSQL. O conhecimento destes podem abrir v´arias portas e deve ser considerado um fator de extrema importância para garantir uma boa empregabilidade. É sempre importante estar atento a novas tecnologias e como elas resolvem problemas provenientes das limitações existentes no caso deste tipo de banco enormes quantidade de dados. Neste tutorial veremos o que vem a ser o banco MongoDB [1] e como proceder sua utiliza¸cão utilizando como pano de fundo a linguagem de programa¸cão Java [2] e Python [3].

1 Parte inicial

MongoDB (de "humongous" - monstruoso) é um Sistema de Banco de dados não relacional, Orientado a Documentos e de fonte aberto. É parte da família de sistemas de Banco de Dados denominados **NoSQL**, ou seja, em vez de armazenar dados em tabelas - como é feito em um banco de dados relacional - armazena seus dados em uma estrutura como JSON, ou seja, documentos com esquemas dinâmicos. Este formato é conhecido como **JSON Binário** ou simplesmente BSON.

Figura 1: *Logo do MongoDB*

Possui como objetivo principal promover uma integração mais fácil e rápida com os dados. E possui as seguintes características:

Escrito em linguagem de programa¸cˆao C++

-
-

Gerenciar coleções de documentos BSON formato de intercâmbio de dados usado principalmente como um formato de armazenamento de dados e transferência de rede no banco de dados MongoDB.
BSON é uma forma bin´aria para a representa¸c˜ao de estruturas de dados simples e matrizes associativas (chamados de objetos ou documentos no MongoDB)

1.1 Criar o contêiner Docker

A forma mais simples de termos o MongoDB é através de um contêiner no Docker, assim facilmente podemos ter várias versões do banco instalada e controlar mais facilmente qual banco está ativo ou não. E ainda colhemos o benefício adicional de não termos absolutamente nada deixando sujeira em nosso sistema operacional ou áreas de mem´oria.

Baixar a imagem oficial:

```
docker pull mongo
```

Criar uma instância do banco em um contêiner:

```
docker run --name meu-mongo -p 27017:27017 -d mongo
```

Acessar o Shell de comandos do MongoDB no contêiner:

```
docker exec -it meu-mongo mongo admin
```

```
> show dbs
> use local
> show collections
> exit
```

Podemos parar o contêiner com:

```
docker stop meu-mongo
```

Ou inici´a-lo novamente:

```
docker start meu-mongo
```

1.2 Shell - a console de comandos

O Mongo Shell, também conhecida como Console de Comandos, utiliza uma interatividade entre comandos JavaScript e o MongoDB. Aqui é possível realizar opera¸c˜oes administrativas como consultas ou manutenç˜oes de dados.

Mostrar as bases de dados existentes:

```
> show dbs
```

Criar (ou mudar) a base de dados para a atual:

```
> use nome base
```

Mostrar as coleç˜oes existentes na base de dados atual:

```
> show collections
```

* **db** deixar como está pois é uma variável interna que aponta para a base de dados atual e **col** deve ser modificada para o nome da coleç˜aonasa¸c˜oesabaixo.

Inserir (ou alterar caso o objeto tenha sido chamado anteriormente) um documento em uma coleção (se a coleç˜aonão existe será criada) na base de dados corrente:
`> db.col.save({"campo1$":"valor1", ..., "campoN":"valorN"})`

Listar os documentos de uma coleç˜ao existente na base de dados atual:
`> db.col.find()`

Listar um documento específico de uma cole¸c˜ao existente na base de dados atual:
`> db.col.find({"campo":"valor"})`

Adicionar uma nova coluna:
`> db.col.update({}, {" set":{"campo":"valor"}}, {upsert:false, multi:false})`

Eliminar documento(s) de uma cole¸c˜ao existente na base de dados atual:
`> db.col.remove({campo:valor})`

Apagar uma coleç˜ao existente na base de dados atual:
`> db.col.drop()`

Apagar a base de dados atual:
`> db.dropDatabase()`

Se percebemos bem a única diferença do MongoDB para bancos relacionais é entendermos como é o relacionamento entre os objetos:

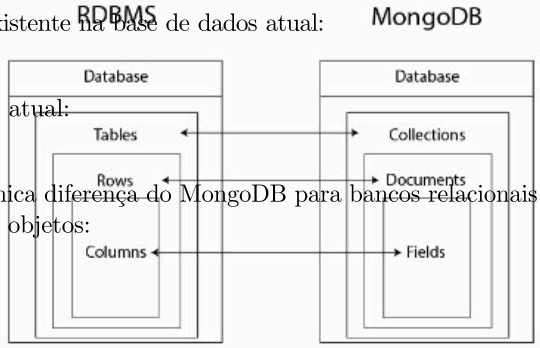

Figura 2: *Comparativo entre os objetos do MongoDB e SQL*

Para conhecer mais comandos do Shell, podemos acessar o seguinte endere¸co: `https://docs.mongodb.org/manual/mongo/`.

2 Linguagem Java

Java é considerada a linguagem de programa¸c˜ao orientada a objetos mais utilizada no Mundo, base para a constru¸c˜ao de ferramentas como Hadoop, Pentaho, Weka e muitas outras utilizadas comercialmente. Foi desenvolvida na década de 90 por uma equipe de programadores chefiada por *James Gosling* para o projeto Green, na Sun Microsystems - tornou-se nessa época como a linguagem que os programadores mais baixaram e o sucesso foi instant^aneo. Em 2008 o Java foi adquirido pela Oracle Corporation.

2.1 Driver JDBC de Conexão

Para proceder a conexão com Java, é necessário baixar um driver JDBC (Java Database Connection). Existem vários drivers construídos, porém o driver oficialmente suportado pelo MongoDB se encontra noendere¸co: `http://mongodb.github.io/mongo-java-driver`

Para utilizar o driver é necessário criar um projeto (vamos usar o **Spring Tool Suite 4**, utilize se quiser qualquer outro editor de sua preferência).

No STS4 acessar a seguinte opção no menu: File ▷ New ▷ Java Project. Informar o nome do projeto (Decus), não esquecer de modificar a opção "Use an environment JRE"para a versão correta da Java Runtime desejada e pressionar o botão Finish. Se está tudo correto teremos a seguinte situação na aba *Project Explorer*:

Figura 3: *Projeto Decus criado*

Vamos convertê-lo para um projeto Apache Maven. Clicar com o botão direito do mouse no projeto e acessar a op¸c˜ao: Configure ▷ Convert to Maven Project. Na janela apenas pressione o bot˜ao *Finish*. Se tudo est´a correto observamos que o projeto ganhou uma letra **M** o que indica agora é um projeto padr˜ao Maven. Ent˜ao foi criado um arquivo chamado **pom.xml**.

Acessar este arquivo e antes da tag BUILD, inserir a tag DEPENDENCIES:

```
<dependencies>
  <!-- Logging -->
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>1.7.5</version>
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.5</version>
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.5</version>
  </dependency>
  <!-- Driver Banco MongoDB -->
  <dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.0.4</version>
  </dependency>
</dependencies>
```

Agora a situaç˜ao do projeto é esta:

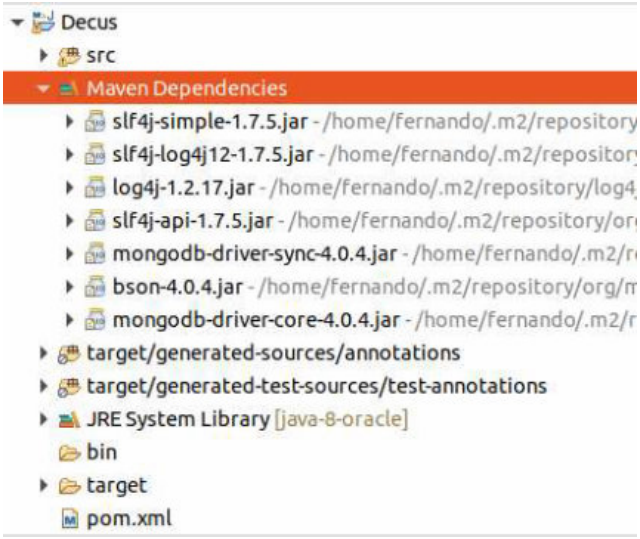

Figura 4: *Dependências do Maven*

Observamos que na pasta **Maven Dependencias** foi baixado a vers˜ao 4.0.4 do driver MongoDB.

2.2 Testar a Conexão

Estamos prontos para testarmos a conexão entre MongoDB e Java. Criamos um pequeno exemplo que nos auxiliará como teste, uma classe chamada **Escola** no pacote **decus.com** e inserimos nesta a seguinte codificaç˜ao:

```
1  package decus.com;
2
3  import org.bson.Document;
4  import com.mongodb.client.MongoClients;
5  import com.mongodb.client.MongoClient;
6  import com.mongodb.client.MongoDatabase;
7  import com.mongodb.client.MongoCollection;
8  import com.mongodb.client.MongoCursor;
9
10 public class Escola {
11   private MongoDatabase db;
12   private MongoClient mongo;
13   private MongoCollection<Document> col;
14
15   protected MongoDatabase getDb() {
16     return db;
17   }
18   protected MongoCollection<Document> getCol() {
19     return col;
20   }
21
22   protected MongoClient getMongo() {
23     return mongo;
24   }
25
```

```
  protected boolean conectar() {
    try {
      mongo = MongoClients.create("mongodb://localhost:27017");
      db = mongo.getDatabase("escola");
      col = db.getCollection("aluno");
    } catch (Exception e) {
      return false;
    }
    return true;
  }

  protected boolean desconectar() {
    try {
      mongo.close();
    } catch (Exception e) {
      return false;
    }
    return true;
  }

  private void executar() {
    if (this.conectar()) {

      // Inserir os alunos
      Document doc = new Document("nome", "Mario da Silva").append("nota",
    (int)(Math.random() * 10));
      col.insertOne(doc);
      doc = new Document("nome", "Aline Moraes").append("nota", (int)(Math.random() *
    10));
      col.insertOne(doc);
      doc = new Document("nome", "Soraya Gomes").append("nota", (int) (Math.random() *
    10));
      col.insertOne(doc);

      // Listar os Alunos
      MongoCursor<Document> cursor = col.find().iterator();
      while (cursor.hasNext()) {
        doc = cursor.next();
        System.out.println(doc.get("nome") + ": " + doc.get("nota"));
      }
      cursor.close();
      this.desconectar();
    }
  }

  public static void main(String[] args) {
    new Empresa().executar();
  }
}
```

Esta classe adiciona três registros ao banco de dados com o nome do aluno e sua nota que é gerada de forma randômica e em seguida procede uma consulta para verificar se os registros foram realmente inseridos. A conex˜ao e desconex˜ao ao MongoDB foi colocada em métodos separados.

No Shell utilizar os seguintes comandos para verificar os dados:

```
> show dbs
```

```
> use escola
> show collections
> db.aluno.find()
```

E se tudo está OK, teremos o seguinte resultado:

Figura 5: *Execuç˜ao do Shell*

2.3 Programação Java usando o MongoDB

Nesta se¸c˜ao ser´a visto como via linguagem Java é possível gerenciar os objetos do MongoDB. Os comandos dos exemplos a seguir foram escritos a partir dos objetos existentes no c´odigo anterior. Por esse motivo deixamos os métodos protegidos ao invés de particulares e criamos os tipo *GET* para objetos que estão na mesma classe.

Criar uma nova classe chamada **TstComando**, que estende a classe **Escola** no mesmo pacote com a seguinte codificaç˜ao:

```
package decus.com;

public class TstComando extends Escola {
  public static void main(String[] args) {
    new TstComando().executar();
  }

  private void executar() {
    if (conectar()) {
      // Inserir o comando aqui
      desconectar();
    }
  }
}
```

Esta classe agora será a nossa principal, sendo assim removemos os métodos **main** e **executar** da classe **Escola** que já serviram a seu propósito. Lembre-se que a Programação Orientada a Objetos é uma metodologia e não uma linguagem, se pratica essa forma ao usarmos os princípios da Orientação a Objetos e aproveitar a qualidade de extensibilidade do c´odigo.

2.4 Informações dos Documentos

Para obter informaç˜oes dos documentos, é possívelutilizardiversasa¸c˜oes.

Listar as bases de dados existentes:

```
for (String s:  getMongo().listDatabaseNames()) {
   System.out.println(s);
}
```

Criar uma nova coleç˜ao na base de dados pelo seu nome:

```
MongoDatabase db2 = getMongo().getDatabase("escola");
```

Verificar quais s˜ao as cole¸c˜oes existentes em uma determinada base de dados:

```
for (String s:  getDb().listCollectionNames()) {
   System.out.println(s);
}
```

Criar uma nova coleção pelo seu nome e através deste obter a quantidade de documentos existentes:

```
MongoCollection<Document> col2 = getDb().getCollection("aluno");
System.out.println("Total de Documentos:"+ col2.countDocuments());
```

Obter, em formato JSON (*JavaScript Object Notation*), as cole¸c˜oes de uma determinada base de dados:

```
ListCollectionsIterable<Document> it = getDb().listCollections();
MongoCursor<Document> cursor = it.iterator();
while (cursor.hasNext()) {
   System.out.println(cursor.next().toJson());
}
cursor.close();
```

Criar um índice para uma coleção, o parâmetro com valor igual a 1 informa que deve ser ordenado de forma ascendente (descendente utilizamos o valor -1):

```
getCol().createIndex(new Document("nota", 1));
```

Obter, em formato JSON, os índices de uma determinada cole¸c˜ao:

```
ListIndexesIterable<Document> it = getCol().listIndexes();
MongoCursor<Document> cursor = it.iterator();
while (cursor.hasNext()) {
   System.out.println(cursor.next().toJson());
}
cursor.close();
```

Eliminar um indice de uma coleç˜ao:

```
getCol().dropIndex(new Document("nota", 1));
```

Obter, em formato JSON, os documentos de uma determinada cole¸c˜ao:

```
MongoCursor<Document> cursor = getCol().find().iterator();
while (cursor.hasNext()) {
   System.out.println(cursor.next().toJson());
}
cursor.close();
```

Para os pr´oximos exemplo, consideraremos o método executar() conforme o c´odigo abaixo e

procedemos a inserç˜ao do comando descrito na posiç˜aoindicada:

```
private void executar() {
  if (conectar()) {
    // Inserir o comando aqui
    while (cursor.hasNext()) {
      System.out.println(cursor.next().toJson());
    }
    cursor.close();
    desconectar();
  }
}
```

2.5 Filtrar Coleções

Limitar a quantidade de documentos retornados (por exemplo 2):

```
MongoCursor<Document> cursor = getCol().find().limit(2).iterator();
```

Trazer os alunos que obtiveram nota 10:

```
MongoCursor<Document> cursor = getCol().find(new Document("nota", 10)).iterator();
```

Através da classe `com.mongodb.client.model.Filters` é possível realizar a mesma aç˜ao:

```
MongoCursor<Document> cursor = getCol().find(Filters.eq("nota", 10)).iterator();
```

E com a utiliza¸c˜ao dessa classe, é possível realizar as seguintes ações:

- **Filters.ne** - registros não iguais a um determinado valor
- **Filters.gt** - registros maiores que um determinado valor
- **Filters.gte** - registros maiores ou iguais a um determinado valor
- **Filters.lt** - registros menores que um determinado valor
- **Filters.lte** - registros menores ou iguais a um determinado valor

Para realizar a mesma consulta com a utilizaç˜ao dos filtros:

```
MongoCursor<Document> cursor = getCol().find(
Filters.and(Filters.gt("nota", 3), Filters.lt("nota", 9))).iterator();
```

Também podemos utilizar as variáveis:

eq - Igual	$
ne - Não igual	
gt - Maior	
gte - Maior ou igual	
lt - Menor	
lte - Menor ou igual.	$ $

Obter todos os documentos da cole¸c˜ao com a nota é maior que 6:

```
MongoCursor<Document> cursor = getCol().find(
new Document("nota", new Document(" gt",6))).iterator();
```

Parece mais complicado, porém é possível criar separadamente um objeto Documento e a partir dele compor combina¸c˜oes. Obter todos os documentos cujas notas s˜ao maiores que 3 e menores que 9:

```
Document doc = new Document();
doc.append("nota", new Document(" gt", 3).append(" lt", 9));
```

```
MongoCursor<Document> cursor = getCol().find(doc).iterator();
```

2.6 Ordenações

Através da classe `com.mongodb.client.model.Sorters`, e podemos utilizar as variáveis "ascending" e "descending" para obter ordena¸c˜oes:

```
MongoCursor<Document> cursor =
col.find().sort(Sorts.ascending("nota")).iterator();
```

3 Modificar os documentos da Coleção via Java

Uma vez identificado o(s) documento(s) desejado(s) é possível proceder:

Altera¸c˜oes.Utilizarosmétodos updateOne ou updateMany.
Eliminações. Utilizar os métodos deleteOne ou deleteMany.

Modificar a nota do aluno "Mario da Silva" para 5:

```
getCol().updateOne(new Document("nome","Mario da Silva"),
new Document(" set", new Document("nota", 5)));
```

Para eliminar o aluno "Mario da Silva":

```
getCol().deleteMany(new Document("nome","Mario da Silva"));
```

3.1 Eliminar os documentos

Para eliminar a cole¸c˜ao "aluno":

```
getCol().drop();
```

Para eliminar a base de dados "escola":

```
getDb.drop();
```

4 Python

Python é uma linguagem de programação de alto nível, interpretada a partir de um script, Orientada a Objetos e de tipagem din^amica. Foi lan¸cada por Guido van Rossum em 1991. N˜ao pretendo nesta apostila COMPARAR essa linguagem com Java (espero que nunca o fa¸ca), fica claro que os comandos s˜ao bem mais f´aceis porém essas linguagens possuem diferentes prop´ositos.

Todos os comandos descritos abaixo foi utilizado no JupyterLab [5], então basta abrir um Notebook e digitá-los em cada célula conforme se apresentam.

4.1 Proceder a Conexão

Baixar o pacote necessário:

```
!pip install pymongo
```

Importar os pacotes necessários:

```
from pymongo import MongoClient
import random
```

Neste caso estamos utilizando o pacote **random** somente para criarmos o mesmo exemplo já visto e escolher uma nota aleatória para casa aluno.

Nos conectamos ao servidor desta forma:

```
cliente = MongoClient('localhost', 27017) Ou:
cliente = MongoClient('mongodb://localhost:27017/')
```

Listar as bases disponíveis:

```
cliente.list database names())
```

Nos conectamos a uma base desta forma:

```
db = cliente.escola Ou:
db = cliente['escola']
```

Listar as coleç˜oesdisponíveis:

```
cliente.list collection names())
```

Conectamos a uma cole¸c˜ao desta forma:

```
col = db.aluno Ou:
col = db['aluno']
```

4.2 Inserir documentos

Inserir um único documento é uma quest˜ao de criar um dicion´ario e enviá-loparaacole¸c˜ao:

```
mario = { "nome":  "Mario da Silva", "nota":  random.randint(1,11) }
col.insert one(mario)
```

Inserir vários documentos é necessário criar uma lista de dicionários e enviar a lista para a coleção:

```
alunos = [
   { "nome":  "Aline Moraes", "nota":  random.randint(1,11) },
   { "nome":  "Soraya Gomes", "nota":  random.randint(1,11) }
]
col.insert many(alunos)
```

4.3 Encontrar documentos

Listar toda a coleç˜ao:

```
for doc in col.find({}):
   print(doc)
```

Listar toda a coleç˜ao de modo ordenado ascendente (ou descendente - valor -1):

```
for doc in col.find({}).sort("campo",1):
   print(doc)
```

Quantos documentos existem na coleç˜ao:

```
col.count documents({})
```

Trazer o primeiro documento:
```
col.find one()
```

Trazer um determinado documento:
```
col.find one({"nome": "Aline Moraes"})
```

Limitar a quantidade de documentos buscados (no caso 5):
```
for doc in col.find({}).limit(5):
  print(doc)
```

Mostrar um determinado campo (e somente ele):
```
for doc in col.find({}):
  print(doc['col'])
```

Trazer os documentos que possuem a nota maior que 5 e menor que 7:
```
for doc in col.find({"nota": {" gt": 5, " lt": 7}}):
  print(doc)
```

4.4 Atualizar documentos

* O lado da esquerda é o filtro de consulta e o lado do SET s˜ao os campos a alterar.

Alterar um documento que possui o nome "Mario da Silva":
```
col.update one({"nome": "Mario da Silva"}, {" set": {"nota": 8}})
```

Alterar os documentos que possuem a nota menor que 5:
```
col.update many({'nota': {' lt': 5}}, {' set': {'nota': 4}})
```

Eliminar um documento que possui o nome "Mario da Silva":
```
col.delete one({"nome": "Mario da Silva"})
```

Eliminar os documentos que possuem a nota menor que 5:
```
col.delete many({'nota': {' lt': 5}})
```

4.5 Encerrar

É boa pr´atica fechar a base de dados:
```
cliente.close()
```

Mas antes de encerramos realmente vejamos o seguinte programa completo em linguagem Python:

```
1 from pymongo import MongoClient
2 from random import randint
3
4 # Passo 1: Conectar ao Mongo
5 cliente = MongoClient(port=27017)
6 db = cliente.negocio
7
8 # Passo 2: Criar Amostras de Dados
9 nomes = ['Kitchen', 'Espiritual', 'Mongo', 'Tastey', 'Big', 'Jr', 'Filho', 'City',
      'Linux',
10          'Tubarão', 'Gado', 'Sagrado', 'Solo', 'Sumo', 'Lazy', 'Fun', 'Prazer', 'Gula']
```

```
tipo_emp = ['LLC', 'Inc', 'Cia', 'Corp.']
tipo_coz = ['Pizza', 'Bar', 'Fast Food', 'Italiana', 'Mexicana',
            'Americana', 'Sushi', 'Vegetariana', 'Churrascaria']

for x in range(1, 501):
  nome1 = nomes[randint(0, (len(nomes)-1))]
  nome2 = nomes[randint(0, (len(nomes)-1))]
  tipoE = tipo_emp[randint(0, (len(tipo_emp)-1))]
  negocio = {
   'nome': nome1 + ' ' + nome2 + ' ' + tipoE,
   'nota': randint(1, 5),
   'cozinha': tipo_coz[randint(0, (len(tipo_coz)-1))]
  }

  # Passo 3: Inserir o objeto negócio no banco
  result = db.restaurante.insert_one(negocio)

  # Passo 4: Mostrar no console o Object ID do Documento
  print('Criado {0} de 500 como {1}'.format(x, result.inserted_id))

  # Passo 5: Mostrar mensagem final
  print('500 Novos Negócios Culin´arios foram criados...')

cliente.close()
```

O programa está auto-documentado e criar uma base com 500 registros.

5 Conclusão

Penso que depois dessa apostila, ser´a possível usar todo o poder do banco MongoDB para seus trabalhos, pois como vimos é bem f´acil realizar os passos nesse banco pouco importa a linguagem de programa¸c˜ao. N˜ao busquei nesta mostrar um exemplo mais completo para n˜ao limitar suas pesquisas e devemos considerar esta apenas como um pontapé inicial (*KickStart*) para seus projetos.

Como visto o banco de dados MongoDB pode ser facilmente utilizado com aplicações em linguagem Java ou gerar os modelos para *Machine Learning* com Python e ainda colher o benefício de substituir os bancos de dados relacionais para grandes quantidades de dados, sendo que esta é a grande motivação para NoSQL como forma de resolver o problema de escalabilidade dos bancos tradicionais.

As principais linguagens de programa¸c˜ao possuem suporte e aqui vimos apenas Java e Python, porém existem muitas outras como PHP, C, C++, C#, JavaScript, Node.js, Ruby, R e Go. Esta apostila faz parte da série dos quatro tipos para Bancos de Dados no padr˜ao NoSQL que estou tentando desmistificar e torn´a-los mais acessíveis tanto para as comunidades de Java e Python voltada especificamente para desenvolvedores ou cientistas de dados.

Figura 6: *Tipos de Bancos de Dados*

Sou um entusiasta do mundo **Open Source** e novas tecnologias. Qual a diferen¸ca entre Livre e Open Source? Livre significa que esta apostila é gratuita e pode ser compartilhada a vontade. Open Source além de livre todos os arquivos que permitem a geração desta (chamados de arquivos fontes) devem ser disponibilizados para que qualquer pessoa possa modificar ao seu prazer, gerar novas, complementar ou fazer o que quiser. Os fontes da apostila (que foi produzida com o LaTex) está disponibilizado no GitHub [8]. Veja ainda outros artigos que publico sobre tecnologia através do meu Blog Oficial [6].

Referências

[1] Página do Banco MongoDB
`https://www.mongodb.org/`

[2] Página do Oracle Java
`http://www.oracle.com/technetwork/java/`

[3] Página do Python
`https://www.python.org/`

[4] Editor Spring Tool Suite para c´odigos Java
`https://spring.io/tools`

[5] Página do Jupyter
`https://jupyter.org/`

[6] Fernando Anselmo - Blog Oficial de Tecnologia
`http://www.fernandoanselmo.blogspot.com.br/`

[7] Encontre essa e outras publica¸c˜oes em
`https://cetrex.academia.edu/FernandoAnselmo`

[8] Repositório para os fontes da apostila
`https://github.com/fernandoans/publicacoes`